VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

PHP Frameworky

PHP Frameworks

2009

Pavel Zbranek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne ....
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Janu Vavříčkovi za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

Abstrakt

Hlavním úkolem této práce je poskytnout dostatečný přehled o možnostech využití, vlastnostech a přínosů frameworků při vývoji webových aplikací v programovacím jazyce PHP. Způsob tvorby webových aplikací dosáhnul dalšího bodu evoluce. Při tvorbě systémů je vhodné se podívat po softwarových strukturách - frameworcích, které nám vývoj usnadňují. V dokumentu je poskytnut potřebný teoretický základ, popsány výhody (nevýhody) a hlavní atributy frameworků, které hrají důležitou roli při jeho výběru. Vybráno je několik frameworků, u kterých jsou popsány jejich vlastnosti a možnosti využití. Vybrané a popsané frameworky v této práci jsou poté taky prakticky použity k tvorbě jednoduchého IS, na kterém jsou demonstrovány jejich nejdůležitější vlastnosti a přednosti. Poté následuje zhodnocení, jak se jednotlivé frameworky zhostily daného úkolu.

Klíčová slova

Autentizace, CakePHP, CodeIgniter, databázová platforma, framework, informační systém, MVC architektura, návrhové vzory, open source, ORM, PHP, Prado, Symfony, validace, webová aplikace.

Abstract

The main goal of this thesis is to provide a sufficient overview of the possibilities of use, characteristics and benefits of frameworks during the development of the web applications in PHP programming language. Development method of the web application reached the next point of evolution. In the development systems it is appropriate to take a look at software structures – frameworks, that faciliate us the development. In document is provide the theoretical background, describe advantages (disadvantages) and the main attributes of frameworks, that play an important role in its selection. Selected are several frameworks where are their properties and posibilities of use described. Selected and described frameworks in this thesis are then also practically used to created the same simple IS, where are demonstrated the most important characteristics and advantages. This is followed by assessment of how individual frameworks has implemented the task.

Keywords

Authentication, CakePHP, CodeIgniter, database platform, framework, information system, MVC architecture, design patterns, open source, ORM, PHP, Prado, Symfony, validation, web application.

Seznam použitých symbolů a zkratek

ACL – Access control lists

AJAX – Asynchronous JavaScript and XML

API – Application programming interface

CMS – content management system

CSS – Cascading style sheets

EDP – Event-Driven Programming

ERP – Enterprise resource planning

HTML – HyperText markup langure

IDE – Integrated Development Environment

JSON – JavaScript Object Notation

MIT – Massachusetts Institute of Technology

MVC – Model View Controller

ORM – Object-relation mapping

PEAR – PHP extension and Application Repository

PDF – Portable document format

PHP – Personal home page

RSS – Really simple syndication

SQL – Structured query language

SOAP – Simple Object Access Protocol

WWW – World wide web

XML – Extensible markup language

Obsah

Seznam příloh

1. Úvod

V dnešní době stále více narůstá množství aplikací vyvíjených v prostředí internetu, nebo intranetu. Ať už se jedná o jednoduché blogovací systémy, e-shopy, nebo kompletní celopodnikové ERP systémy, jsou nároky na tyto aplikace stále rostoucí. Už menší webová aplikace se může stát během vývoje velmi nepřehlednou. Na různých místech se opakují stále stejné části kódů řešící stejný problém. A jen následná údržba takto navrženého systému může být velmi problematická a finančně nákladná. Právě tyto výše popsané problémy lze řešit pomocí softwarových struktur, které nám pomáhají při tvorbě softwaru. Obsahuji knihovny řešící známé problémy, nebo API. Stejně tak doporučené a osvědčené návrhové vzory, které nás vedou správným směrem při tvorbě systému. Takto navržený systém je přehledný a především usnadní práci v týmu.

Tato práce si ovšem neklade za cíl pouze vyjmenovat a popsat všechny dostupné frameworky pro jazyk PHP. Vybráno je pouze několik frameworků, které budou důkladně popsány a ukázáno na nich, jak se s nimi pracuje a jaké možnosti nabízí.

Cílem této práce je tedy popsat výhody a nevýhody, které nám plynou z použití frameworků při vývoji internetových aplikacích v jazyce PHP. Budou popsány vlastnosti a funkce, které můžeme u frameworků pro jazyk PHP nalézt, stejně tak náročnost, složitost nastavení a způsob zakomponování do vyvíjeného systému u vybraných frameworků. Způsob jejich použití bude ilustrován na jednoduchém, ukázkovém systému.

2. PHP frameworky

Před samotným popisem jednotlivých frameworků a jejich vlastností bude důležité si vysvětlit, co to vůbec frameworky jsou a základní principy jejich použití. Co je jejich úkolem při tvorbě moderních internetových aplikacích, jaké součásti může obsahovat a jak se liší vývoj aplikací pomocí frameworku oproti klasickému vývoji.

Aplikace vyvíjené pro prostředí internetu, nebo intranetu jsou odlišné a liší se projekt od projektu. Proto není vhodné podcenit výběr vhodného frameworku, který bude uspokojovat naše požadavky. Je tudíž velmi důležité se zorientovat ve vlastnostech a technologiích, které jednotlivé frameworky pro vývoj poskytují.

2.1. Co je to framework

Framework je základní konceptuální struktura, která je použita k řešení nebo určení komplexních problémů[1]. Tato definice je ovšem příliš všeobecná a pro naše účely pochopení frameworků v oblasti vývoje software nám moc neporadí. Tyto struktury jsou využívány v mnoha oblastech a proto se v následujících kapitolách při používání pojmu framework budeme zaměřovat pouze na činnosti z oblasti softwarového inženýrství.

2.2. Aplikační framework

V oblasti informačních technologiích a programování se aplikační framework, nebo taky webový aplikační framework, může označovat za softwarovou strukturu, která je použita pro implementování osvědčených struktur, postupů, návrhových vzorů apod. Při vývoji aplikací v prostředí internetu je tedy webový aplikační framework softwarový framework, který je navržen pro podporu vývoje, dynamických webových stránek webových aplikací a webových služeb[2].

Bez těchto aplikačních struktur je tvorba systémů v prostředí internetu velmi náročná. Současné webové stránky jsou složitější a obsahují mnohem více aplikační logiky. Frameworky si tedy kladou za cíl usnadnit vývoj a následnou údržbu těchto systémů. Učinit projekt transparentní a jednotný pro všechny členy, kteří se na vývoji podílí. Snaží se popsat nejlepší postupy a činnosti, které se u vývoje webových aplikací stále opakují a můžou se použít na další projekty. Nejčastějšími nástroji a funkcemi, které frameworky obsahují, jsou návrhové vzory, knihovny API[1], šablony, osvědčené postupy apod. Nejčastější požadavky na framework je jeho jednoduché použití a nasazení. Stejně tak doba potřebná pro jeho naučení a pochopení. Často se frameworkům vytýká, že pro jejich nastudování je potřeba mnoho času, ale při jejich opakovaném použití se rychlost vývoje rapidně zkracuje. V dnešní době je ale asi nejdůležitější požadavek, aby framework implementoval architekturu, která při vývoji odděluje model dat od aplikační logiky a uživatelského rozhraní. Tuto podmínku dnes splňuje architektura návrhového vzoru MVC (Model-View-Controller), která bude detailněji popsána v této kapitole.

[1] API - Application programming interface. Je rozhraní pro programování aplikací. Obsahuje funkce a knihovny, které může programátor použít při programování aplikací.

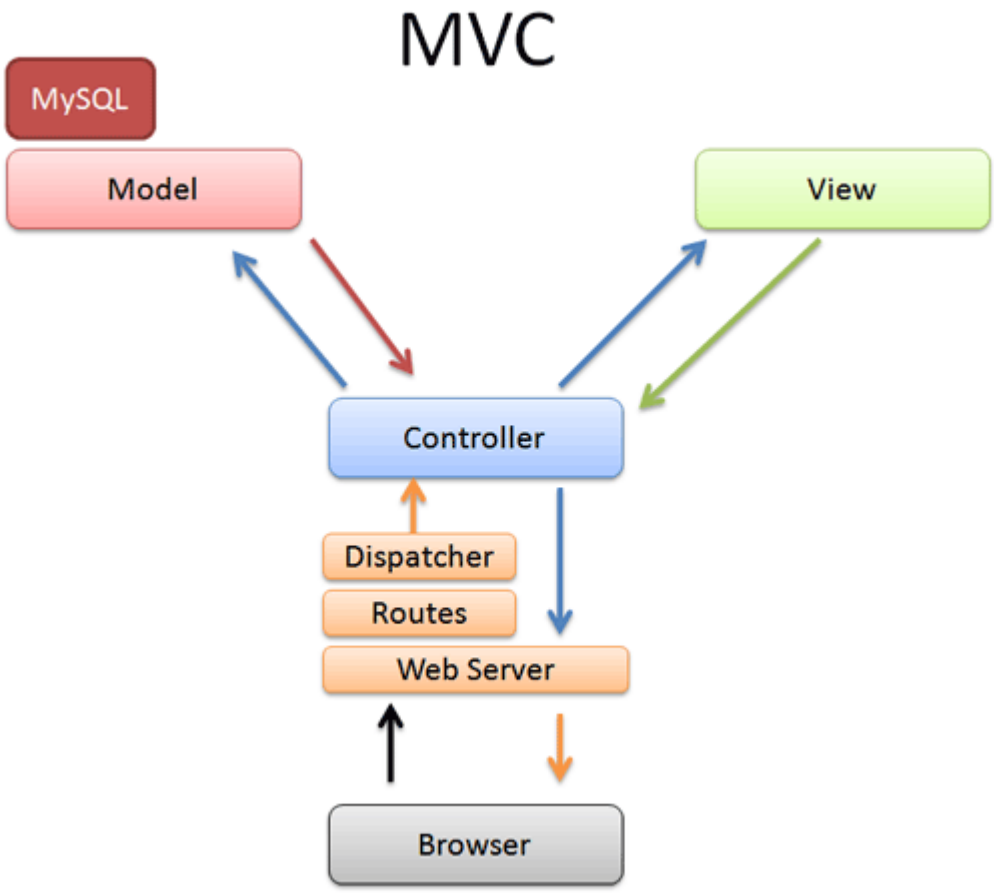
2.3. Programovací jazyk PHP a frameworky

PHP je skriptovací programovací jazyk, který je určen převážně k tvorbě dynamických webových stránek. Webové aplikace jsou rozdílné a proto také frameworky pro tento jazyk budou mít rozdílné vlastnosti a funkce. Všechny frameworky pro tento jazyk jsou open source[2]. Žádný z těchto frameworků není oficiálním pro jazyk PHP a většinou vznikaly jako iniciativa komunit okolo tohoto oblíbeného jazyka. Při jejich tvorbě se tvůrci často snažili svůj framework a jeho principy stavět podle existujících frameworků pro jiné jazyky, například framework Ruby on Rails a podle něj postavený framework CakePHP.

Výčet všech frameworků pro programovací jazyk PHP je velice obsáhlý. Existují jak velké a zavedené frameworky, tak i malé za kterými stojí jeden člověk. Poměrně aktuální přehled o nejpoužívanější frameworcích a jejich základních vlastnostech je k nalezení na internetu[3].

Všeobecně se dá také říci, že velmi důležitou věcí každého frameworku je kvalitní dokumentace. PHP framework je tak kvalitní, jak silná je komunita lidí okolo něj.

2.4. MVC Framework



Obrázek 1: Schéma MVC modelu[4]

[2] Open source – Počítačový software s otevřeným zdrojovým kódem.

Asi nejdůležitější vlastností PHP frameworků je implementace architektury MVC (Model-View-Controller). Základní myšlenkou je oddělení logiky aplikace, datového modelu a prezentace dat v aplikaci. Při vývoji se rozdělení uskutečňuje pomocí tří, na sobě nezávislých, komponent. Úplná nezávislost těchto komponent je pouze relativní. Výraznější změna jedné komponenty se vždy částečně promítne do druhé.

2.4.1. Model

Model (model) je vrstva, která reprezentuje data (informace) aplikace – datová vrstva aplikace. Ve většině frameworků pro jazyk PHP model reprezentuje jednotlivé tabulky databáze. Každá tabulka databáze by tedy měla mít svůj model, který ji reprezentuje. Proto taky všechny činnosti spojené s vložením, editací, mazání jsou umístěny právě v modelu.

2.4.2. View

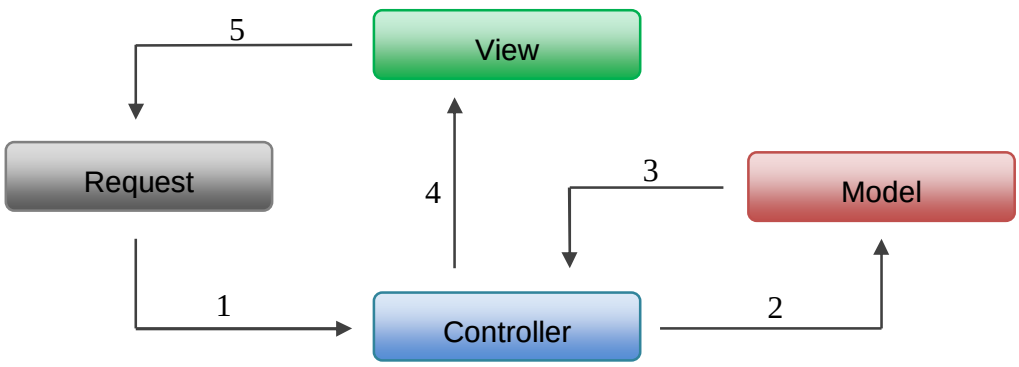
View (pohled) se stará o poskytnutí rozhraní pro prezentování dat uživateli – prezentační vrstva. Stejně tak mu umožňuje zadávání požadavku. Vytváří výstupy a šablony nad daty, které mu předá vrstva Model. Většinou se jedná o HTML s vepsaným PHP kódem, ale taky to můžou být XML, PDF a další soubory vhodné pro prezentaci dat.

2.4.3. Controller

Controller (ovladač) představuje hlavní a nejdůležitější část architektonického vzoru MVC, který řídí komunikaci mezi zbylými dvěma komponentami. Obsahuje logiku aplikace, řídí její provoz a obsluhuje požadavky od uživatele – vrstva řízení logiky aplikace. Controller by měl jenom předávat požadavky a akce Modelu.

2.4.4. Princip fungování MVC

Nyní, když známe funkci jednotlivých komponent, si vysvětlíme a popíšeme, jakým způsobem probíhá komunikace mezi nimi při jednoduchém požadavku:



Obrázek 2: Princip komunikace mezi jednotlivými komponentami

1) Zaslaný požadavek Controlleru od uživatele s uživatelskými daty

2) Controller zpracovává požadavek a posílá žádost Modelu pro přístup k datům

3) Model odpovídá požadavku od Controlleru a posílá mu zpátky data nebo zprávu o uložení informací.

4) Controller posílá výstupní data Pohledu

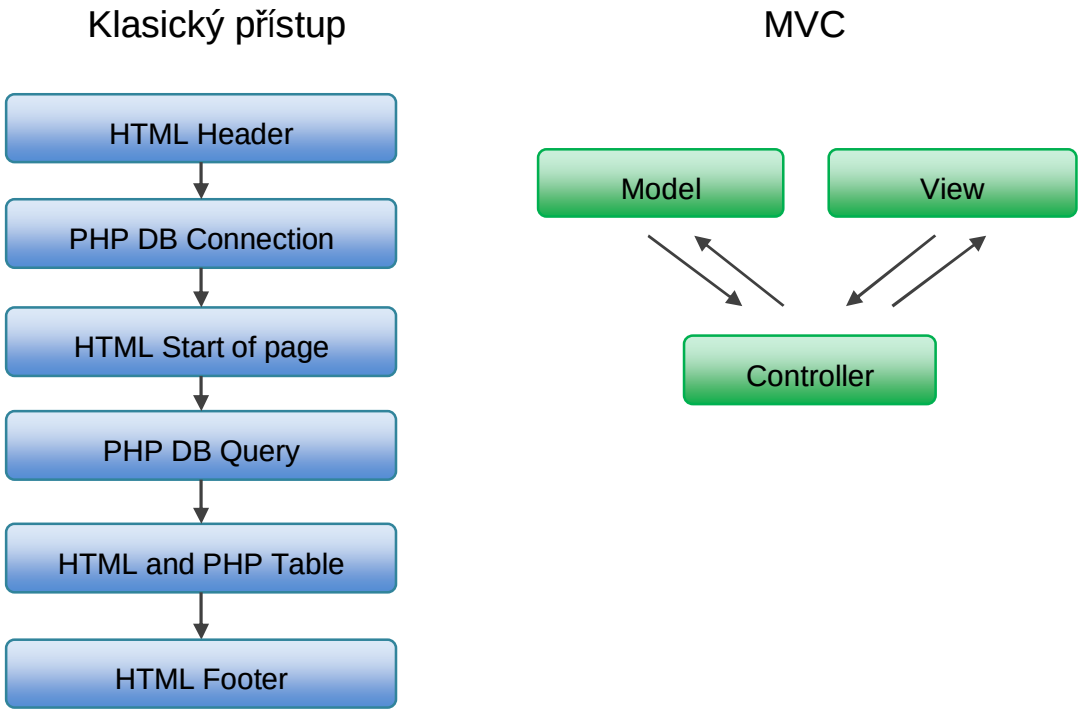
5) View vypíše data v požadovaném formátu

Ze způsobu komunikace vidíme nezávislost jednotlivých komponent. Díky tomu můžeme dělat potřebné změny mnohem rychleji. Vývoj celé aplikace je standardizován pod jednotnou architekturou. Proto je nasazení frameworků výhodné zejména při práci ve vývojových týmech. Každý člen týmu ví přesně, kde hledat svou práci, ať už se jedná návrháře, programátora anebo správce databáze[5].

2.5. Důvody použití PHP frameworků v internetových aplikacích

2.5.1. Běžné programování v jazyce PHP

Klasicky napsaný program v jazyce PHP je HTML kód, do kterého jsou na různých místech vpisovány bloky kódů v jazyce PHP. Takto vytvořená struktura PHP stránky je velmi nepřehledá. I v případě, že je použito funkcí jazyka PHP pro vkládání segmentů kódů, které jsou umístěny ve vyhrazených souborech, není logika aplikace, uživatelský výstup a ani přístup k datům v databázi oddělen v komponentách.



Obrázek 3: Rozdíl mezi klasickým přístupem a MVC architekturou

2.5.2. Výhody použití PHP frameworků

Výhody plynoucí z používání frameworků byly již částečně popsány na začátku kapitoly. Pokusíme se tedy jen o jakousi sumarizaci.

- Přehledná práce v týmu
- Jednotná architektura projektu
- Rychlý vývoj a údržba
- Přehledný návrh – HTML vypadá jako HTML
- Znovupoužitelnost kódu
- Provádění jednotkových testů
- Množství balíčků pro práci (většinou neinvazivní[3])
    - Mail
    - Generování PDF
    - Vyhledávání na webu
    - Ajax
- Zjednodušení řešení při vývoji
    - Správa session
    - Autentizace a autorizace
    - Jazyková lokalizace
    - Tvorba RSS kanálu

2.5.3. Nevýhody použití PHP frameworků

Na samotném nasazení PHP Frameworku příliš mnoho nevýhod nenalezneme. Spíše by se mohlo jednat o nevhodné použití daného frameworku pro konkrétní projekt. Stejně tak se nabízí otázka, zda vůbec framework pro projekt nasadit. V případě že se jedná o jednoduché stránky prezentující například firmu, tak bude nasazení frameworku zbytečné a třeba bychom se měli raději podívat po nějakém vhodném šablonovém řešení - například Texy[6]

Někdy se taky frameworkům vytýká jejich složitost, delší doba pro naučení a s tím spojený pomalý vývoj aplikací. Ten je ovšem pouze v počátcích používání. S postupem času je vývoj s jejich pomocí mnohem rychlejší.

2.6. Atributy PHP frameworků

Většina frameworků se od sebe liší vlastnostmi a funkcemi, které nabízí. Proto si uvedeme ty hlavní vlastnosti, kterými se frameworky prezentují a hrubě si popíšeme jejich význam pro vyvíjenou aplikaci

- PHP4/PHP5 – Verze jazyka PHP, kterou frameworky podporují. Téměř všechny podporují verzi 5. Některé i starou verzi 4 zřejmě z důvodů kompatibility u starších projektů

- MVC – Návrhový vzor (popsáno v úvodu kapitoly)

[3] Neinvazivní - znamená, že se můžou jednotlivé balíčky použit samostatně a na ničem nezávisí. Zbytek frameworku nemusíme vůbec používat.

- Více databází – Podpora více DB bez nutnosti jakékoliv změny
- ORM – podpora Object-Relational Mapping
- Databázové objekty – podpora dalších databázových objektů
- Šablony – framework má v sobě zabudovaný šablonový systém
- Cachování – Podpora uchování stránek do vyrovnávací paměti
- Validace – obsahuje funkce pro validaci nebo filtrování komponent
- Ajax – Obsahuje podporu jazyka Ajax ve frameworku.
- Autentizace – obsahuje funkce pro obsluhu autentizace přihlašujících uživatelů
- Moduly – Obsahuje moduly jako podpora e-mailu, PDF, RSS, …
- EDP – Událostmi řízené programování

3. Vybrané PHP frameworky

3.1. Výběr frameworků

PHP frameworků je v současné době k nalezení velké množství. Kromě důležitých vlastností a funkcionalit, které daný framework nabízí, se zaměříme na další důležité požadavky. Při výběru testovaných frameworků mělo asi nejdůležitější roli rozšíření používání daného frameworku. Nemělo by smysl vybírat framework, jehož vývoj se dávno zastavil a budoucí podpora dalšího vývoje je nejistá. Stejně jako malý, nebo nový framework pokrývající pouze některé části vývoje webových aplikací a okrajová řešení některých problémů. Souhrn nejrozšířenějších PHP frameworků je k dispozici na webové stránce *www.phpframeworks.com*, kde jsou přehledně v tabulce zaznamenány jejich nejpodstatnější vlastnosti.

Vybrané frameworky by měly být komplexní a ve většině případu pokrývající veškeré typické činnosti při vývoji aplikací v prostředí internetu. Většinou se jedná o frameworky, které jsou používány pro větší projekty, jejich vývoj pokračuje delší dobu a stojí za nimi silná komunita vývojářů, nebo firma, která framework vyvíjí. V případě, že nad nějakým frameworkem stavíme systém je velmi důležitá další jeho podpora a vývoj. Důležitá je také kvalitní dokumentace, popis a způsob použití funkcí.

Jedním z nejdůležitějších a největších frameworků pro programovací jazyk PHP je Zend Framework. Ten je vyvíjen firmou Zend Technologies Ltd. a stojí za ním početná komunita i velcí partneři – Google, IBM, Microsoft, Adobe a další. Tento Framework je ovšem až příliš obsáhlý. Proto bude z našeho výběru vynechán a bude popsán v samostatné práci.

3.2. CakePHP

CakePHP je vysoce produktivní webový aplikační framework usnadňující vývoj aplikací v jazyce PHP. Samotný framework je rovněž napsán v jazyce PHP. Vznikat začal roku 2005 a jeho autorem je Michal Tatarynowicz. Od začátku byl inspirován frameworkem Ruby on Rails. Umožňuje psát robustní aplikace strukturovaným a rychlým způsobem bez ztráty flexibility. Jeho další předností je téměř žádná nutnost nastavování.

Základní údaje:

- Vyvíjeno: Cake Software Foundation, Inc.
- Vznik: 2005
- Aktuální verze: 1.2.0.7962
- Licence: MIT license[4]
- Webové stránky: www.cakephp.org

[4] MIT license – Svobodná licence, která vznikla na Massachusetts Institute of Technology. Je velmi podobná BSD licencí a dovoluje se software nakládat téměř neomezeně. Podmínkou je zahrnutí textu licence do všech kopií odvozených od původního software.

Vlastnosti CakePHP:

- Žádná konfigurace – pouze nastavení údajů databáze
- Kompatibilní s PHP4 a PHP5
- Flexibilní licence
- Automatické generování (Administrace systému)
- MVC architektura
- Integrovaná součinnost s databází pro CRUD[5] a zjednodušené dotazy
- Dobře vypadající uživatelské URL
- Rychlé a flexibilní šablony
- Podpora Pohledu pro AJAX, JavaScript, HTML formuláře a další
- Email, Cookie, bezpečnost, správa session, komponenty pro obsluhu požadavku
- Validace dat
- Flexibilní Caching Pohledu
- Podpora ACL[6]
- Jazyková lokalizace aplikace
- Podpora jednotkových testů využívající *Simple test framework*[7] (od verze 1.2) [8]

Dokumentace a uživatelská podpora:

Komunita okolo toho frameworku je početná a proto i zdrojů, ze kterých je možno čerpat, je dostatek. Na oficiálních stránkách frameworků je detailně zpracován popis API. Dále pak manuál, který provází uživatele při práci s tímto frameworkem od úplných začátků. Ke konci manuálu je přidán vývoj ukázkové aplikace. Knih v anglickém jazyce je také dostatek.

3.3. Symfony

Symfony je framework pro tvorbu opravdu rozsáhlých aplikací a dává vývojářům plnou kontrolu nad nastavením. Z tohoto důvodu je často využíván firmami pro vývoj robustních celopodnikových systémů. Za jeho vznikem a podporou stojí francouzská firma Sensio Labs. Tento framework je postaven na jazyce PHP5 a stejně tak podporuje vývoj systémů pouze pro PHP5.

Základní údaje:

- Vyvíjeno: Sensio Labs
- Vznik: 2005
- Aktuální verze: 1.2.1
- Licence: MIT license
- Webové stránky: www.symfony-project.org

[5] CRUD – Create: vytvořit nový záznam, Read: vypsat nebo vyhledat položky, Update: aktualizace, editace položky, Delete: smazání existujícího záznamu. Jsou čtyři základní funkce pro práci s relačními databázemi.
[6] ACL - Access control lists. Seznam řízení přístupu – seznam určuje kdo, nebo co bude mít povolení přistupovat k objektům a jaké bude moct provádět operace s tímto objektem.

Vlastnosti Symfony:

- Složitější konfigurace
- Několik možností instalací a správa projektů z příkazové řádky
- Zdrojové knihovny frameworku může využívat více aplikací najedou
- Kompatibilní s většinou databází
- Postaveno na známých projektech (Prado, Mojovi, Propel)
- Možnost vlastního rozšíření Symfony, nebo pomocí pluginů
- MVC architektura
- Pěkně vypadající URL
- Generátory – Administrace, CRUD
- Debugování
- Cachování
- Validace formulářů
- Podpora AJAX
- Podpora lokalizace a jazykových překladů aplikace
- Integerovaný testovací framework LIME (Funcional, Unit) [9]

Dokumentace a uživatelská podpora:

Kvalitních zdrojů, které by byly dostupné v češtině, téměř není. Výjimkou je částečný překlad úvodních kapitol oficiální dokumentace. Proto jsou hlavním zdrojem informací oficiální stránky projektu, kde se nachází dokumentace. Tento framework je aktivně vyvíjen pouze rok a tudíž i aktivní komunita okolo toho frameworku není příliš početná. Největším nedostatkem pro studium tohoto frameworku jsou téměř žádné návody, ukázky řešení typických problémů znázorňující postup při vytváření komplexní aplikace postavené nad frameworkem Symfony.

3.4. Prado

Prado je komponentový, událostmi řízený webový aplikační framework. Podporuje jazyk PHP 5.1.0 a vyšší. Základní myšlenky, na kterých je stavěn, si vypůjčuje z ASP.NET frameworku. Jeho zakladatelem je Qiang Xue. Framework Prado prošel dlouhým vývojem, který začal již v roce 2004. Od současné verze je již tento framework použitelný pro vývoj větších webových aplikací. I tento framework, stejně jako předešlé, je open source.

Základní údaje:

- Vyvíjeno: PRADO Group
- Vznik: 2004
- Aktuální verze: 3.1.3
- Licence: BSD License[7]
- Webové stránky: www.pradosoft.com

Vlastnosti Prado:

[7] BSD License – Představuje volnou softwarovou licenci. Umožňuje komerční nasazení, či libovolné další použití. Nutno je uvést autora a informaci o licenci.

- Objektově orientovaný a znovupoužitelný kód
- MVC architektura
- Událostmi řízené programování
- Kompatibilní s většinou databází
- Cachování – statické ukládání komponent, které byly inicializované
- HTML komponenty
- Podpora webových služeb[8] (JSON a SOAP)
- Nastavení projektu pomocí souboru XML
- Webové komponenty desktopových aplikací – validace, původce nastavením
- Podpora lokalizace a jazykových překladů aplikace
- Přizpůsobitelná obsluha chyb a vyjímek
- Podpora dokumentace
- Autentizace a autorizace
- Podpora jazyka AJAX
- Kontrola CAPTCHA[9] u formulářů
- Pokročilé zabezpečení webových aplikací [10]

Dokumentace a uživatelská podpora:

Na oficiálních stránkách je napsaná dokumentace. Dále pak dokumentace API a popis tříd, které Prado nabízí. Jako hlavním zdrojem při studiu tohoto frameworku můžou být tutoriály, které ukazují tvorbu jednotlivých funkcionalit až po komplexní internetové systémy jako je blog. Jeden z výukových materiálů je dostupný volně na stránkách Prado frameworku jako kniha ve formátu pdf.

3.5. CodeIgniter

CodeIgniter je webový aplikační framework, který umožňuje psát velice rychle a efektivně aplikace v jazyce PHP téměř bez nutnosti jakéhokoliv nastavování. Podporuje jazyk PHP 4.3.2 a novější. Tím je zaručena bezproblémová funkčnost, pokud internetovou aplikaci provozujeme na hostovaném serveru. Tím, že je tento framework menší, běží stránky v něm vytvořené mnohem rychleji, než u ostatních frameworků. Je velmi jednoduchý na používání a proto velmi usnadní práci při vytváření nejen menších aplikací. Autorem tohoto frameworku je Rick Ellis.

Základní údaje:

- Vyvíjeno: EllisLab, Inc.
- Vznik: 2006
- Aktuální verze: 1.7.0
- Licence: BSD license
- Webové stránky: www.codeigniter.com

[8] Webová služba – Umožňuje aplikaci volat pomocí protokolu HTTP funkce, které jsou umístěné na vzdáleném serveru. Nezáleží přitom na použité technologii, protože je použito formátu XML pro výměnu dat a parametrů.
[9] CAPTCHA - *C*ompletely *A*utomated *P*ublic *T*uring test to tell *C*omputers and *H*umans *A*part – Automatický test pro odlišení počítačů a lidí. Používá se především pro ověřování u vyplňování formulářů. Test spočívá zpravidla v zobrazení obrázku s deformovaným textem, přičemž úkolem uživatele je zobrazený text opsat do příslušného vstupního pole.[11]

Vlastnosti CodeIgniter:

- Podpora většiny databází – přístup pomocí Active records
- MVC architektura
- Správa seancí
- Cachování
- Validace dat formulářů
- Tvorba hezkých URL
- Zabezpečení
- Podpora jazykové lokalizace
- Funkce – jednouchá práce s obrázky, Excel, captcha, generátory PDF, email,
- Podpora jednotkových testů [12]

Dokumentace a uživatelská podpora:

Přestože je tento framework relativně mladý je komunita uživatelů, právě pro jeho jednoduchost, veliká. Největším zdrojem informací je kvalitně a přehledně vytvořený manuál na oficiálních stránkách frameworku. Dále pak několik video tutoriálů, které provedou základním nastavením a používám tohoto frameworku a ukážou jeho možnosti na ukázkové aplikaci. Na těchto stránkách je taky velmi užitečný dokument, který popisuje pravidla, jak správně psát aplikaci postavenou nad tímto frameworkem. Dále je k dispozici jako u většiny frameworků fórum, nebo wiki stránky, které jsou plné rad a hotových řešení.

3.6. Srovnání vlastností PHP frameworků

Většina frameworků se příliš neliší v různých implementovaných vlastnostech, protože se snaží komplexně pokrýt potřeby pro tvorbu webových aplikací, a proto jejich funkcionality jsou téměř totožné. Pro srovnávací tabulku byly proto vybrány především vlastnosti, ve kterých se vybrané frameworky liší.

	PHP4	PHP5	ORM	Ajax	AuthModule	EDP
CakePHP	✓	✓	✓	✓	✓	
Symfony		✓	✓	✓	✓	
Prado		✓	✓	✓	✓	✓
CodeIgniter	✓	✓				

4. Zhodnocení frameworků na ukázkovém systému

4.1. CakePHP

4.1.1. Instalace frameworku CakePHP

Framework CakePHP je distribuován ke stažení jako komprimovaný balíček ve formátech bz2, gz a zip. Tento předpřipravený balíček po stažení rozbalíme do kořenového adresáře našeho webového serveru. Zde nám vznikne nový adresář reprezentující nový projekt. Je pojmenován podle označení aktuální verze (např. *1.2.x.xxxx*), a proto si vytvořený adresář přejmenujeme libovolně podle názvu projektu např. *NazevProjektu*.

Pokud se podíváme do adresáře vytvořeného projektu, měl by mít následující strukturu:

```
/webroot/NazevProjektu /app       - adresář app obsahuje soubory a složky
                                    vyvíjené aplikace.
                       /cake      - obsahuje hlavní knihovny frameworku,
                                    které CakePHP aplikace využívá. Obsah
                                    toho adresáře nás v podstatě nezajímá.
                       /docs      - adresář docs obsahuje soubory naší
                                    dokumentace, licenční podmínky apod.
                       /vendors   - obsahuje knihovny a funkční kody třetích
                                    stran
```

Požadavky na platformu:

Pro použití frameworku je důležité správné prostředí. Pro spuštění je třeba mít nainstalovaný skriptovací jazyk PHP minimálně verzi 4.3.2. Všechny pozdější verze PHP by měly pracovat bez problémů. Samozřejmostí je taky webový a databázový server pro úplné využití možností aplikace. Nejčastěji se používá webový server Apache a databázový server MySQL. CakePHP také funguje s ostatními webovými a databázovými servery.

Již nyní si můžeme otestovat správnou funkčnost prostředí a nastavení frameworku zadáním adresy *http://localhost/NazevProjektu/* do internetového prohlížeče. V případě, že je vše v pořádku by se měla objevit úvodní stránka CakePHP, která nás informuje o vlastnostech nastavení a prostředí.

4.1.2. Nastavení frameworku CakePHP

Před zahájením používání musíme provést několik drobných nastavení prostředí a frameworku CakePHP. Jedná se o nastavení webového serveru Apache a nastavení pro práci frameworku s databázovým serverem.

Nastavení webového serveru Apache:

Nastavení AllowOverwrite – musíme zkontrolovat, zda je parametr AllowOverwrite kořenového adresáře webového serveru nastaven na hodnotu *all*. Toto nastavení provedeme

v konfiguračním souboru ***http.conf***[10], který je umístěn v adresáři ***conf***, kde je Apache nainstalován.

```
<Directory "c:/web/www">
     Options Indexes Includes FollowSymLinks MultiViews
     AllowOverride All
     Order allow,deny
     Allow from all
</Directory>
```

Zapnutí modu mod_rewrite – Povolení modu mod_rewrite se provádí stejně jako výše v konfiguračním souboru http.conf. Zde se pouze odstraní komentář na řádku, kde se po mod_rewrite povoluje.

```
LoadModule rewrite_module modules/mod_rewrite.so
```

Nastavení CakePHP frameworku pro práci s databázovým serverem:

Jako první si musíme nejdříve vytvořit novou databázi na SŘBD, kterou bude aplikace využívat. Pojmenujeme ji třeba ***nazevprojektu.*** Nyní se nastaví CakePHP pro práci s databázovým serverem MySQL. Otevřeme si adresář ***\NazevProjektu\app\config*** a soubor ***database.php.default*** přejmenujeme na ***database.php.*** Nyní v tomto souboru nastavíme parametry pro připojení k databázi, které jsou umístěny v následujícím poli.

```
var $default = array(
            'driver' => 'mysql',
            'persistent' => false,
            'host' => 'localhost',
            'login' => 'root',
            'password' => 'naseheslo',
            'database' => 'nazevprojektu',
            'prefix' => '',
);
```

4.1.3. Adresářová struktura projektu

Všechny soubory a složky projektu jsou umístěny v adresáři ***app.*** Adresářová struktura projektu ve frameworku CakePHP je relativně složitá a proto popíšeme vlastnosti pouze u základních adresářů, nebo případně adresářů, které jsou podstatné pro vývoj základní aplikace.

[10] Pojmenování konfiguračních souborů a adresářů odpovídá pro instalaci webového serveru Apache2 pod operačním systémem Microsoft Windows.

```
app/config/                 -obsahuje konfigurační soubory pro DB, acl ...
  /controllers/             -obsahuje vytvořené soubory kontrolerů
  /controllers/components/  -vlastní nadefinované kompomemty
  /locale/                  -obsahuje soubory pro jazykovou lokalizaci
  /models/                  -obsahuje soubory modelu
  /plugins/                 -obsahuje soubory rozšiřujících modulů
  /tests/                   -obsahuje soubory pro testy
  /tmp/                     -obsahuje soubory pro cachování a log soubory
  /vendors/                 -obsahuje programové kody třetích stran
  /views/                   -soubory pohledu pro zobrazení údajů
        /errors/            -vlastní chybové stránky
        /layouts/           -soubory pro rozvržení dokumentu
        /pages/             -obsahuje obyčejný text pto zobrazení
  /webroot/                 -kořenový adresář webové aplikace
          /css/             -soubory CSS pro definování vzhledu stránek
          /files/           -ostatní soubory
          /img/             -obrázky
          /js/              -soubory jazyka JavaScript
```

4.1.4. Způsob práce s frameworkem a konstrukce MVC na příkladu

Pro lepší představu, jak se s tímto PHP frameworkem pracuje, si ukážeme jednoduchý příklad. Bude se jednat o výpis dat z tabulky a vložení jednoho záznamu. Hlavní pro nás bude, jakým způsobem se implementuje návrhový vzor MVC a které náležitosti každý vytvářený soubor musí splňovat. Tento framework není nutné v podstatě vůbec nastavovat a začít s ním je jednoduché. Na druhou stranu od nás vyžaduje množství konvencí, které musíme dodržovat.

Návrh Modelu:

Jak již bylo řešeno v úvodu, je model zodpovědný za přístup a modifikaci dat v tabulce databáze. A proto každá tabulka databáze by měla být reprezentována svým jediným modelem.

V adresáři */NazevProjektu/app/models/* vytvoříme nový soubor *uzivatel.php*. Jméno tabulky by mělo být uvedeno v množném čísle, zatímco jméno modelu v jednotném čísle. Tyto pravidla pojmenování jsou povinné konvence, které si framework vynucuje pro jeho správný chod. Další informace o konvencích pro pojmenování objektů jsou uvedeny v kapitole zhodnocení toho frameworku. Součástí modelu je také velmi jednoduše implementována validace vstupních dat.

Obsah souboru *uzivatel.php:*

```
<?php
      class Uzivatel extends AppModel {
      var $name = 'Uzivatel';
      var $validate = array(
           'prijmeni' => array(
```

```
            'rule' => 'alphaNumeric',
            'required' => 'true',
            'message' => 'Příjmení uživatele nemůže být prázdné'
        )
    );
    }
?>
```

Implementace Controlleru:

Controller slouží k řízení a obsluze příchozích požadavků. Po přesunutí do adresáře */NazevProjektu/app/controllers/* zde vytvoříme soubor *uzivatele_controller.php*. Každý Controller implementuje jednotlivé metody. Tento jednoduchý Controller pouze implementuje metodu *index()* pro výpis dat a metodu *add()* pro zápis nového záznamu.

Obsah souboru *uzivatele_controller.php:*

```
<?php
class UzivateleController extends AppController {
    var $name = 'Uzivatele';
    function index() {
        $this->set('uzivatele', $this->Uzivatel->find('all'));
    }
    function add() {
        if (!empty($this->data)) {
            $this->Uzivatel->create();
        if ($this->Uzivatel->save($this->data)) {
            $this->Session->setFlash('Nový uživatel byl uložen');
            $this->redirect(array('action'=>'index'), null, true);
        } else {
            $this->Session->setFlash('Uživatel nebyl uložen!');
        }
    }
}
?>
```

Implementace View:

Pohled slouží k prezentování dat uživateli, stejně jako pro zadávání požadavků. Můžeme tedy jeho pomocí vypsat obsah tabulky uložené v databázi anebo sbírat data prostřednictvím formuláře. Přemístíme se tedy do složky */NazevProjektu/app/views/* vytvoříme uvnitř nový adresář pojmenovaný jako *uzivatele*. Uvnitř tohoto adresáře budeme ukládat pohledy související s *Uzivatele*. Vytvořený pohled pojmenujeme *index.ctp*. Je nutné dávat pozor na koncovku pohledu, protože všechny pohledy ve frameworku CakePHP mají koncovku .ctp

Obsah souboru *index.ctp* pro zobrazení obsahu tabulky:

```
<h2>Uživatelé</h2>
```

```
<?php if(empty($uzivatele)): ?>
     Žádný uživatel k zobrazení
<?php else: ?>
<table>
     <tr>
         <th>Title</th>
         <th>Status</th>
         <th>Created</th>
         <th>Actions</th>
     </tr>
     <?php foreach ($uzivatele as $uzivatel): ?>
        <tr>
          <td>
            <?php echo $uzivatel['Uzivatel']['jmeno'] ?>
           </td>
           <td>
            <?php echo $uzivatel['Uzivatel']['prijmeni'] ?></td>
          </td>
          <td>
            <?php echo $uzivatel['Uzivatel']['mesto'] ?>
          </td>
        </tr>
     <?php endforeach; ?>
</table>
<?php endif; ?>
```

Obsah souboru *add.ctp* pro zobrazení formuláře Přidat uživatele:

```
<?php echo $form->create('Uzivatel');?>
    <fieldset>
        <legend>Přidat nového uživatele</legend>
        <?php
          echo $form->input('jmeno');
          echo $form->input('prijmeni');
          echo $form->input('mesto');
        ?>
    </fieldset>
<?php echo $form->end('Add Task');?>
```

4.1.5. Implementace zajímavých funkcí ve frameworku CakePHP

Většina frameworků nabízí základní funkce, které jsou většinou shodné. Od aplikování návrhového vzoru MVC, zjednodušenou validaci vstupních formulářů, nebo různými Helpery usnadňující práci s formuláři, emaily apod. Důležitým faktorem při implementaci určitého frameworku může být sada zajímavých a nadstandardních vlastností, které framework nabízí.

Tvorba RSS kanálu[11]:

Velice zajímavou a užitečnou vlastností, která je implementována na dnešních moderních webových stránkách, je možnost se přihlásit k odebírání aktuálních příspěvků a zpráv pomocí RSS kanálu. Framework CakePHP tuto vlastnost nabízí k implementaci. Tvorba RSS kanálu je standardně dostupná od verze 1.2.xxxx, ve starších verzích pouze pomocí rozšiřujícího modulu.

Postup vytvoření RSS kanálu:

1) Vybrání elementu, který chceme přidat k RSS čtečce
2) Vytvořit šablonu pro RSS/XML pohled
3) Vytvořit samotný pohled, který obsahuje funkci *rss_transform*

Vytvoření nového RSS kanálu pro webovou stránku je velice jednouché a především rychlé. Celá procedura přidání kanálu trvá pouze pár minut[13].

Automatické generování administrace:

Framework CakePHP nám umožňuje automatické vygenerování všech potřebných stránek a formulářů pro obsluhu databázové tabulky s funkcemi pro vložení, editaci, detail a smazání záznamů. Tato funkce je ve frameworku pojmenována jako *scaffolding* – lešení. Pojmenování atributů a popisu vygenerované tabulky se děje taky automaticky a je shodné s názvy atributů v databázi. Tato funkce je velice užitečná především při vývoji aplikace, pokud potřebujeme jednoduše spravovat obsah databázových tabulek, nebo se stále očekávají změny ve struktuře databáze. Používání automaticky vygenerované správy obsahu databáze se ovšem doporučuje pouze v období vývoje. Pro nasazení v produkčním prostředí by se měly všechny části implementovat manuálně.

Nastavení je jednouché a spočívá pouze v přidání členské proměnné *$scaffold* do příslušného kontroléru, který je vázán na model, respektive tabulku databáze.

Ukázka vygenerování formulářů a tabulek pro model Uzivatel:

```
<?php
   class UzivateleController extends AppController {
      var $scaffold;
   }
?>
```

Testování pomocí jednotkových testů:

Jednotkové testy jsou při vývoji aplikací velmi důležité. Hlavním principem jednotkových testů je testování po menších samostatných jednotkách, metodách, funkcích, kdy se porovnává očekávaný výsledek se skutečným výsledkem funkce. Ovšem jen velmi zřídka se s nimi setkáme v případě malých systémů, projektů, které jsou vyvíjeny jedním člověkem, který má nad systémem úplný přehled. V případě větších systémů a při práci je nutnost techniky testování používat. Čím dál častěji se také setkáváme s vývojem řízeným testováním *(Test Driven Development)*. U jazyků, které

[11] RSS kanál je množina formátů XML sloužící pro čtení novinek a nových zpráv webových stránek. Uživatel je po přihlášení okamžitě upozorněn na novou zprávu, či komentář.

nemají datové typy a nejsou čistě objektové, je jednotkové testování nezbytné. Framework CakePHP nabízí integraci testovacího balíčku *SimpleTest* pro jednotkové testy v jazyku PHP. Stažený testovací framework musíme nainstalovat do složky /vendors. Všechny vytvořené testy se provedou vždy najednou.

Ukázka jednotkových testů ve frameworku CakePHP:

```
<?php
require_once('simpletest/unit_tester.php');
require_once('simpletest/reporter.php');

function spoctiCenu($cenaKnihy, $dph, $sleva) {
  $soucet = $cenaKnihy * (($dph/100)+1);
  $soucet = $soucet + ($sleva/100)*$soucet;
}

class TestCeny extends UnitTestCase {
  function testSpoctiCenu() {
    $this->assertEqual(714, spoctiCenu (500, 19, 20), 'Text...');
  }
}
$test = &new TestCeny();
$test->run(new HtmlReporter());
?>
```

Jazyková lokalizace stránek:

Lokalizace systémů je dnes díky mezinárodnímu prostředí velmi důležitá a to především u systémů v prostředí internetu. Implementace systému pro podporou více jazyků nemusí být vždy triviální, a proto možnost jednoduchým způsobem lokalizovat stránky a zvýšit tím komfort práce se systémem pro uživatele je velmi zajímavá a může hrát důležitou roli při výběru daného frameworku.

Pro nastavení lokalizace musí každý kontrolér obsahovat na začátku import třídy pro lokalizaci. Nejlepší řešení je uvést tuto deklaraci v kontroleru appController, který rozšiřuje AppController. Poté nebudeme muset deklaraci uvádět v každém kontroleru:

```
App::import('Core', 'l10n');
```

Vytvoříme soubory s lokalizací pro jednotlivé jazyky. Musíme ovšem zadávat zkratku lokalizace podle ISO normy pro názvy zkratek jakzyků[14]:

```
locale/eng/LC_MESSAGES/default.po (English)
locale/cze/LC_MESSAGES/default.po (Czech)
```

Naplníme soubory s názvem identifikátoru a text:

```
msgid  "vypln"
msgstr "Vyplňte formulář"
msgid  "ukoncit"
msgstr "Zavřít objednávku"
```

4.1.6. Zhodnocení

Jednou z největších předností Frameworku CakePHP vidím především v téměř žádném nastavování před samotným použitím. Po rozbalení frameworku a nastavení parametrů databáze můžeme okamžitě začít na tvorbě aplikací. Je jednoduchý a jeho hlavním cílem je opravdu zjednodušit práci a pokrýt rutinní práce při vývoji. Přesto je ovšem množinou svých funkcí rozsáhlý a umožňuje jednoduše integrovat do internetových stránek většinu vlastností, které jsou u informačních systémů v prostředí internetu, nebo intranetu obvyklé. Další velmi důležitou věcí je dostatek knih, kvalitních materiálů pro studium a hotových návodů na řešení obvyklých problémů v tomto frameworku.

Jako nevýhodu bych viděl ve velmi přísných konvencích pojmenování objektů, ve kterých se ztrácí přehled. Kontrolér, model, pohled, funkce a další názvy mají přesně definované pravidla pro psaní prvního písmena velkým písmenem, nebo množného čísla názvu. V případě množného čísla automaticky očekává písmeno „s“ na konci slova. Tyto konvence ovšem CakePHP vyžaduje také na databázové úrovni pro pojmenování databázových tabulek a dokonce některých atributů. Tuto skutečnost vidím jako velký nedostatek především v případě, že se aplikace vyvíjí již nad existující databází, nebo dokonce nad databází, se kterou již spolupracují ostatní programy.

Tento framework se může použít pro tvorbu malých, ale klidně i středně velkých systémů ať už se jedná o internetový obchod, CMS systém, nebo menší podnikový ERP.

4.2. Symfony

4.2.1. Instalace frameworku Symfony

Možnosti instalace u frameworku Symfony jsou rozmanitější, než u ostatních frameworků. Právě díky těmto a mnoha dalším vlastnostem může být Symfony vhodné především velké a rozsáhlé projekty. K dispozici jsou tři paralelně vyvíjené větve frameworku.

Možnosti instalace:

1) První možnost instalace je klasickým způsobem stáhnout ze stránek instalační soubory a ty všechny rozbalit do kořenového adresáře našeho webového serveru. Instalační balíčky jsou k dispozici ke stažení ve formátu *.tgz* a *.zip*.Tato možnost se označuje v dokumentaci a uživatelské příručce jako *sandbox*.

2) Druhou možností je instalace nového projektu pomocí SVN / Subversion ze vzdáleného úložiště, které je umístěno na stránkách frameworku. Tato možnost je doporučována pokročilejším uživatelům[20].

   Příkaz z příkazové řádky pro vytvoření projektu:

```
> svn checkout http://svn.symfony-
project.com/tags/RELEASE_1_0_0/
```

3) Poslední možností je instalace pomocí funkce (frameworku) PEAR, který je od verze 4.3.0 součástí jazyka PHP. PEAR je distribuční systém pro projekty v jazyce PHP.

Příkaz pro instalaci frameworku Symfony:

```
> pear channel-discover pear.symfony-project.com
> pear install symfony/symfony
```

O správnosti instalace frameworku se můžeme přesvědčit pomocí vložení níže uvedeného odkazu do internetového prohlížeče. Měla by se zobrazit úvodní stránka, která nás informuje o správnosti instalace a dalších krocích nastavení.

```
http://localhost/nazev_projektu/web/index.php/
```

Framework Symfony, stejně jako Prado, umožňuje mít na produkčním serveru nainstalované knihovny a další zdrojové soubory pouze na jediném místě disku serveru. Toto jádro frameworku je posléze využíváno všemi běžícími aplikacemi. Popsaný postup se doporučuje především z důvodu bezpečnosti před napadením jádra frameworku, kdy není možno přistupovat k jádru přímo z internetu.

4.2.2. Nastavení frameworku Symfony

Průběh nastavování nového projektu nepovažuji za příliš jednoduché a přehledné. Celkově lze framework označit jako ten náročnější na nastavení a následné studium. Při novém projektu nelze pouze nastavit základní funkcionality, jak tomu je zvykem u ostatních porovnávaných frameworků, ale téměř během celého vývoje se neobejdeme bez opakovaných konfigurací.

Veškeré nastavování probíhá pomocí příkazové řádky operačního systému. Je tudíž nezbytné nastavit systémové proměnné jazyka PHP. Tímto způsobem za nás framework vygeneruje spoustu věcí, ale je nutná znalost všech příkazů. Mnohem lepší a přehlednější průběh nastavování považuji pomocí konfiguračních souborů. Ostatní frameworky stejně většinou, kromě nastavení databáze, nevyžadují od uživatele velké množství nastavení.

Způsob, jakým framework Symfony přistupuje k nastavování je velice propracovaný a na první pohled je jasné, že tento přístup je zvolen především pro potřebu velkých projektů. U ostatních frameworků je zvykem, že každý vytvořený projekt (instalace) odpovídá jedné vytvořené aplikaci. Symfony umožňuje vygenerovat více aplikací (frontend, backend, …) pro jeden projekt. A takto je uzpůsobené nastavení, které je pro celý projekt, nebo zvlášť nastavení pro každou aplikaci. Veškeré konfigurační soubory jsou v jazyce YAML, což je značkovací jazyk, který je podobný XML. V případě potřeby je možnost psát konfigurační soubory v jazyce XML.

Nastavení práce s databázovým serverem:

Bez práce s databází se neobejdeme a proto jako první inicializujeme datový model. Framework Symfony má k dispozici celkem dva ORM systémy – Doctrine a Propel. Přednastavený pro použití je automaticky Propel. Inicializace datového modelu probíhá pomocí jazyka YAML, což je velice jednoduchý jazyk, který dovoluje XML zápis pomocí stromové struktury. Vytvoříme tedy v adresáři *nazev_projektu/config* soubor, který se bude jmenovat *schema.yml* a vložíme do něj zápis datového modelu. Jména tabulek odpovídají později vygenerovaným třídám.[21]

Ukázka zápisu datového modelu v souboru *schema.yml*:

```
propel:
  movies:
    id:                ~
    nazev_cesky:        { type: varchar(255), required: true }
    nazev_puvodni:      { type: longvarchar }
    popis:             { type: longvarchar }
    zanr:              { type: longvarchar }
    zeme:              { type: longvarchar }
    delka:             { type: longvarchar }
    dabing:            { type: longvarchar }
    titulky_ceske:     { type: longvarchar }
  coments:
    … dále podobně pro všechny zbývající tabulky
```

Datový model vygenerujeme z příkazové řádky pomocí příkazu:

```
> php symfony propel:build-model
```

Nyní musíme přeměnit schéma do příkazů SQL pro vytvoření databázových tabulek. Standardně je nastaven framework Symfony pro práci s jednoduchými SQLite soubory. Framework tak používá databázi uloženou v souboru *sandbox.db*. Pokud chceme pro nový projekt použít třeba databázi MySQL musíme to opět provést pomocí příkazu příkazové řádky.

```
> php symfony configure:database "mysql:dbname=symfony_project;
host=localhost" root heslo
```

4.2.3. Adresářová struktura projektu

Pokud si zvolíme možnost vytvoření nového projektu pomocí možnosti rozbalení zabaleného balíčku od kořenového adresáře webového serveru, tak pak by měl mít nově vytvořený projekt následující strukturu

```
sf_sandbox/
  /apps/             -obsahuje složku pro každou aplikaci v projektu. Často
                     jako frontend a backend
  /apps/frontend/
  /cache/            -ukládá cachovací soubory pro urychlení aplikace
  /config/           -obsahuje základní nastavení aplikace
  /data/             -obsahuje datové soubory jako databázové schéma a
                     soubory SQL pro vytvoření databáze
  /doc/              -zahrnuje soubory dokumentace projektu
```

```
  /lib/              -obsahuje cizí knihovny a třídy, které budeme
                     v projektu využívat
  /log/              -do adresáře framework Symfony ukládá logovací soubory
  /plugins/          -obsahuje nainstalované programy
  /test/             -obsahuje jednotkové a funkcionální testy napsané
                     v PHP, které jsou kompatiblní s testovacím frameworkem
                     v symfony
  /web/              -kořenový adresář pro webový server. Jediné soubory,
                     které jsou přístupné z internetu
  /web/css/          -obsahuje CSS soubory projektu
  /web/images/       -obsahuje obrázky projektu
  /web/js/           -obsahuje soubory JavaScript
```

Protože je ale možné ve frameworku Symfony vytvářet více aplikací pro jeden projekt, tak si ještě popíšeme adresářovou strukturu jedné aplikace.

```
sf_sandbox/apps/frontend/
  /config/           -adresář obsahuje konfigurační soubory aplikace .yml
  /i18n/             -soubory pro internacionalizaci
  /lib/              -obsahuje externí knihovny
  /modules/          -všechny moduly dané aplikace
  /templates/        -šablony (soubory prezenční vrstvy)
```

4.2.4. Způsob práce s frameworkem a konstrukce MVC na příkladu

Návrh Modelu:

Model je základní část každé aplikace a ve frameworku Symfony je reprezentována pomocí ORM nástroje Propel. V kapitole nastavení jsme si již ukázali, jakým způsobem se nadefinuje zápis struktury datového modelu. Po provedení příkazu pro vytvoření souborů, které budou reprezentovat datovou vrstvu naší aplikace, se v adresáři */NazevProjektu/lib/* vytvořil nový adresář */model/* se vygenerovaly třídy, které nám umožňují přístup k relační databázi bez pomoci dotazovacího jazyka SQL.

Nyní provedeme vygenerování sql souboru pro inicializaci databáze podle popisu struktury datového modelu, který je uložen v souboru *schema.yml*.

```
> php symfony propel:build-sql
```

Nyní pomocí příkazu vytvoříme strukturu databázových tabulek pomocí sql souboru, který jsme si vygenerovali v předešlém kroku.

```
> php symfony propel:insert-sql
```

Jako poslední krok potřebujeme vygenerovat třídy podle struktury modelu, které nám umožní řídit objekty modelu. Vygenerované třídy se nacházejí v adresáři */NazevProjektu/lib/form*.

```
> php symfony propel:build-forms
```

Vytvoření aplikace:

Symfony umožňuje vytvořit základní aplikaci bez napsání téměř žádného řádku kódu pouze pomocí příkazů. Tento postup vygenerování všech souborů pro základní aplikaci je doporučený. Posléze se vygenerované soubory upravují podle potřeby.

Příkazy pro vygenerování aplikace:

```
> php symfony propel:generate-module --non-verbose-templates --with-show
frontend user AppUser

> php symfony propel:generate-module --non-verbose-templates frontend
comment AppComment

> php symfony cache:clear
```

Pomocí těchto příkazů jsme vytvořili dva nové moduly (user a comment), které jsou umístěné v adresáři */NazevProjektu/apps/frontend/modules/*. Moduly obvykle slouží k reprezentování stránky, nebo skupiny stránek se stejným cílem. Tyto moduly jsou dostupné pomocí URL:

```
http://localhost/nazev_projektu/web/frontend_dev.php/user
http://localhost/nazev_projektu/web/frontend_dev.php/comment
```

Návrh pohledu:

Automaticky vytvořená aplikace vytvořila moduly. Každý modul se skládá z akcí a jim příslušným šablon – soubory prezenční vrstvy. Pojmenování šablon má danou jmennou strukturu, proto když vytvoříme akci index.class.php, tak příslušný pohled musí být pojmenován indexSuccess.php. Tato struktura modulu vychází z architektury MVC. Automaticky vygenerované pohledy jsou umístěné v adresáři */NazevProjektu/apps/frontend/templates/*.

4.2.5. Implementace zajímavých funkcí ve frameworku Symfony

Framework Symfony implementuje většinu standardních funkcí, které jsou k nalezení u většiny ostatních frameworků. Tento framework ovšem prezentuje jako svou největší výhodu právě práci s frameworkem z příkazové řádky pomocí níž můžeme automaticky vygenerovat části vytvářené aplikace.

Automatické generování administrace systému:

Většina frameworku nabízí možnost generování rozhraní pro základní operace nad tabulkou databáze (vložení, zápis, editace a smazání záznamu). Tato funkce bývá označována jako *scaffolding* – lešení. Její použití je ovšem doporučené pouze v průběhu vývoje aplikace a vhodné jej použít pro produkční nasazeni. Funkce generující aplikaci administrace je v Symfony doporučována i pro použití v provozu. Typicky se takový typ aplikace projektu pojmenovává jako *background* a slouží administrátorovi pro jednoduchou správu dat v databázi.

Postup vygenerování administračního rozhraní:

Předpokládá se, že je nový projekt nainstalován, nastaven a především je nastavena práce s databází pomocí ORM nástroje Propel.

1) Zadání příkazu k vytvoření administračního rozhraní pro danou tabulku
```
> php symfony propel-init-admin backend user User
```
Příkazu *propel-init-admin* náleží parametry: <Název_aplikace>, <Název_modulu>, <Název_třídy>

Tento příkaz vytvoří modul user s akcemi (create, edit, delete a list) podle třídy Propelu User.

2) Příkaz provedeme pro každou tabulku databáze, kterou chceme do administračního rozhraní zahrnout.
3) V adresáři */apps/backend/modules/user/config/* nalezneme soubor *generator.yml*. Toto je vygenerovaný konfigurační soubor administračního rozhraní pro danou tabulku, který můžeme měnit a upravovat tak administrační systém pro tabulku našim potřebám.

4.2.6. Zhodnocení

Symfony framework je vyvíjen už dlouho dobu soukromou firmou Sensio, která se specializuje na webové technologie a disponuje silným zázemím nejen pro další vývoj, ale především podporu existujících projektů, které jsou nad frameworkem postaveny. Jako příklad by se mohl uvést známé aplikace Yahoo Bookmarks, nebo další z celosvětově nejpoužívanějších systému pro tvorbu záložek del.icio.us. Symfony je robustní a komplexní PHP framework určený pro opravdu velké projekty, na němž pracuje rozsáhlý vývojový tým.

Jako jeho největší nevýhodu bych viděl v komplikovaném nastavování v průběhu tvorby celého projektu. Navíc všechny tyto konfigurační procesy se uskutečňují z příkazové řádky, kdy je nutno znát všechny příkazy. Těmito příkazy za nás framework sice nastaví a vytvoří automaticky spoustu věcí, ale na druhou stranu zase nemáme úplnou kontrolu nad generovanými soubory, a tak trošku je uživateli skryt princip jak framework uvnitř vlastně funguje a méně do jeho fungování vidí, což může být na škodu. Konfigurace pomocí upravení údajů v konfiguračních souborech vidím jako vhodnější a přehlednější způsob.

Jako hlavním zdrojem při studiu tohoto frameworku poslouží především jeho stránky, kde je několik průvodců studiem, kteří ukazují, jak postupovat při tvorbě základní aplikace. Dále pak velmi obsáhlá kniha, která už rozebírá všechny aspekty frameworku. Celkově bych tento framework hodnotil spíše jako ten složitější jak na studium, tak i následný vývoj.

4.3. Prado

4.3.1. Instalace frameworku Prado

Po stažení obsahuje celý balíček mnohem více věcí, než samotné jádro frameworku. Obsahuje také nástroje pro testování webových aplikací SimpleTest a Selenium. Dále ještě podporu frameworku Prado pro IDE nástroje a adresář *demos* obsahující velké množství aplikací ukázky aplikací, kde každé demo pokrývá určitou specifickou oblast vývoje webových aplikací. Tyto ukázky můžou výrazně ulehčit seznámení a studium tohoto frameworku. Instalace tohoto frameworku může probíhat dvěma způsoby. Tím prvním může být, jako ve většině případů, pouze nakopírování obsahu balíčku do kořenového adresáře webového serveru, vytvoření adresářové struktury projektu a promazání případně nepotřebných adresářů. Tou druhou cestou může být použití příkazu příkazové řádky pro automatické vygenerování a umístění projektu.

Zajímavou vlastností Prado frameworku je, že jádro samotného frameworku nemusí být nutně umístěno na webovém serveru, ale kdekoliv na disku serveru. V nastavení tak pouze uvedeme cestu ke frameworku. Tento způsob umožňuje provozovat více webových aplikací nad jediným jádrem frameworku.

Adresářová struktura frameworku:

```
/webroot/NazevProjektu /demos     - adresář demos obsahuje více jak desítku
                                    aplikací jako ukázku prací nad
                                    frameworkem Prado.
                       /editors   - Obsahuje rozšíření některých IDE o
                                    podporu tohoto frameworku.
                       /framework - jádro celého frameworku.
                       /requirements- kontrola webového serveru, zda jeho
                                    nastavení vyhovuje požadavků Prado.
                       /tests     - Obsahuje dva testovací frameworky pro
                                    webové aplikace
```

Požadavky na platformu:

Požadavky na platformu jsou standardní jako webový server podporující PHP5. Dále pak podporuje většinu používaných databázových serverů. Pokud jsme nakopírovali celý obsah balíčku na webový server, tak již nyní můžeme spustit úvodní stránku frameworku s manuálem a ukázky programů.

4.3.2. Nastavení frameworku Prado

V případě, že použijeme vygenerování nového projektu pomocí příkazové řádky, tak nám jakékoliv základní nastavování odpadá. V případě, že nový projekt nakopírujeme, tak budeme muset v hlavním adresáři projektu v souboru *index.php* nastavit cestu, kde je framework umístěný a adresář, kde budou umístěny naše soubory.

```
$basePath=dirname(__FILE__);
$frameworkPath=$basePath.'\framework\prado.php';
$assetsPath=$basePath.'\assets';
$runtimePath=$basePath.'\protected\runtime';
```

Další případné nastavení aplikace provedeme v souboru *application.xml*, který se nachází v adresáři *protected*. Většina základních věcí se dá provést bez jeho nastavení a tak může zůstat tento soubor prázdný. Protože vzhled webových aplikací je tvořen pomocí CSS souborů a framework Prado podporuje tvorbu pomocí šablon (mají koncovku .tpl) nastavíme v souboru pages/config.xml cestu adresáře, kde se css soubor s dalšími soubory, jako obrázky a podobně, nachází. Každý další adresář, který se nachází v adresáři *pages* by měl zase obsahovat soubor *config.xml*. Pomocí něj určujeme pravidla zabezpečení a přístupu k jeho obsahu[18].

Nastavení práce s databázovým serverem:

U většiny aplikací se ale nevyhneme práci s databázovým serverem, a proto bude potřeba ještě nastavit připojení k námi vybrané databázi. V případě, že bychom nenastavili přístup k databázi v konfiguračním souboru, tak bychom museli u každého přístupu k databázi uvádět vytvoření databázového spojení, které je velmi podobné jako u čistého jazyka PHP.

```
$dsn = mysql:host=localhost; dbname=filmy
$conn = new TDbConnection($dsn, 'uzivatel','heslo');
TActiveRecordManager::getInstance()->setDbConnection($conn);
```

Nastavení databázového spojení je ovšem doporučováno nastavit pouze na jednom místě v konfiguračním souboru *application.xml* nebo *config.xml*. Tag <*module*> pro nastavení databáze může vypadat následovně. Celkově mi ale přijde práce s Active Record[12] ve frameworku Prado obtížnější a méně intuitivní, než v ostatních frameworcích.

```
<modules>
     <!-- Nastavení přístupu do databáze - DAO a Active Record -->
     <module class="System.Data.ActiveRecord.TActiveRecordConfig"
      EnableCache="true">
```

[12] Active Record – je návrhový vzor používaný pro ukládání dat v relačních databázích. Je to přístup pro připojení k datům v databázi, kdy databázové tabulky, nebo pohledy jsou zabaleny do třídy. Tímto způsobem je svázána instance objektu se záznamem v tabulce[19].

```
          <database ConnectionString="mysql:host=localhost; dbname=filmy"
           Username="uzivatel" Password="heslo" />
     </module>
</modules>
```

4.3.3. Adresářová struktura projektu

Vytvořena adresářová struktura projektu je přehledná a jednoduchá. Pokrývá ale pouze základní činnosti, a proto bude potřeba v průběhu práce s frameworkem dodatečně soubory a adresáře přidávat.

Vhodná struktura by mohla vypadat následovně:

```
app/assets/                  -adresář pro veřejně publikované soubory
  /protected/comon/          -obsahuje další soubory šablon vkládané a
                             hlavní šablony
  /protected/pages/          -zahrnuje soubory alpikace jako pohledy a
                             controléry a stránky se statickým obsahem
  /protected/runtime/        -využívá se frameworkem pro ukládání dat během
                             povozu aplikace, jako cache soubory apod
  /themes/                   -obsahuje soubory css souborů, obrázků a
                             dalších pro definování vzhledu
```

4.3.4. Způsob práce s frameworkem a konstrukce MVC na příkladu

V případě frameworku Prado je základem pohled, který je reprezentován souborem s koncovkou *.page*, která může obsahovat HTML prvky a především dostupné komponenty frameworku. Ke každé takovéto stránce je posléze vytvořen kontrolér, který obsahuje metody pro obsluhu komponent.

Návrh Modelu:

Ve frameworku Prado je každá databázová tabulka reprezentována třídou, která rozšiřuje třídu *TActiveRecord* a dědí od ní tedy všechny implementované metody sloužící pro manipulaci s daty uloženými v tabulce. V části pro deklaraci musí být uveden přesný název tabulky a seznam deklarací všech atributů tabulky, které musí přesně odpovídat skutečné struktuře v databázi. V případě, že je tabulka ve vztahu s jinými tabulkami se ještě musí implementovat statická metoda pro definování vztahu s konkrétní tabulkou.

Je docela vhodné pro přehlednost aplikace všechny vytvořené třídy ActiveRecod, které reprezentují v aplikaci tabulky databáze, uložit do předem vytvořeného souboru určeného pouze pro třídy Active Record. Vytvořený adresář se může pojmenován např. *database*. Tento soubor se ovšem

musí zaregistrovat v konfiguračním souboru *application.xml*, protože framework nemá o jeho existenci ve struktuře projektu žádné tušení, a proto by nemohl využívat soubory v něm obsažené.

Nastavení registrace souboru s třídami Active Record:

```
<paths>
    <using namespace="Application.database.*" />
  </paths>
```

Ukázka třídy Active Record *userRecord.php* v adresáři *nazevProjektu\protected\database*:

```
<?php
    class UserRecord extends TActiveRecord {

        const TABLE='users';

        public $id;
        public $jmeno;
        public $prijmeni;
        public $titul;
        public $pohlavi;
        public $mesto;
        public $telefon;
        public $mail;
        public $spravce;
        public $login;
        public $heslo;

        public static function finder($className=__CLASS__) {
            return parent::finder($className);
        }
    }
?>
```

Implementace Controlleru:

Controller musí být implementován pro každý pohled a musí být také stejně jako pohled pojmenován pouze s rozdílem koncovky, která musí být oproti pohledu ve tvaru ***php***. Vytvořené třídy jsou pak umístěny v adresáři *nazevProjektu\protected\pages* a dále logicky členěných podadresářích (například všechny pohledy a controllery obsluhující tabulku s uživateli by měly být umístěny v jenom adresáři). V controlleru se implementují funkce pro obsluhu požadavku od pohledu. Každá třída, která implementuje daný controller, musí rozšiřovat třídu ***TPage.*** Výsledná třída controlleru se pak velmi podobá třídám z objektově orientovaných programovacích jazyků, kde na začátku jsou uvedeny deklarace atributů třídy a posléze metody včetně konstruktoru, který se ve frameworku Prado označuje *onInit($param)*.

Ukázka controlleru, který předá pohledu úplný seznam uložených uživatelů v databázi a dále ještě prázdnou metodu pro obsluhu stisku tlačítka. Obsah souboru *AdminUser.php:*

```
<?php
      class AdminUser extends TPage {

            public $user;

            public function onInit($param) {

                  parent::onInit($param);
                  $this->user=UserRecord::finder()->findAll();
                  if($this->user===null) {
                        throw new THttpException(500,'Nebyl nalezen
                        záznam');
                  }
                  $this->Title = $this->user->jmeno;

                  parent::onLoad($param);
                  if(!$this->IsPostBack) {
                        $this->DataGrid->DataSource = $this->user;
                        $this->DataGrid->dataBind();
                  }
            }

            public function buttonClicked($sender,$param) {

            }
      }
?>
```

Implementace View:

Pro prezentaci dat uživateli slouží u pohledu ve frameworku Prado obsáhlá sada komponent. Sada komponent je opravdu velmi obsáhlá, a proto určitý čas zabere jejich studium a způsob použití. Použití základních komponent je jednoduché, ale velká část komponent je složitých a jejich použití není příliš intuitivní. Oproti většině ostatních frameworků probíhá validace vstupních dat formulářů právě v pohledu. Všechny soubory reprezentující pohled ve frameworku Prado musí mít koncovku *.page*.

Obsah souboru *AdminUser.page* pro zobrazení obsahu tabulky uživatelů:

```
<com:TContent ID="page">
    <h3>Správa uživatelů</h3>

      <com:TDataGrid
```

```
            Width="670px"
            CellPadding="1"
            ID="DataGrid"
            AutoGenerateColumns="false"
            HeaderStyle.BackColor="black"
            HeaderStyle.ForeColor="white"
            ItemStyle.BackColor="#BFCFFF"
            AlternatingItemStyle.BackColor="#E6ECFF">

            <com:TBoundColumn
                  HeaderText="Jméno"
                  DataField="jmeno"
            />

            <com:TBoundColumn
                  HeaderText="Příjmení"
                  DataField="prijmeni"
            />

            <com:TBoundColumn
                  HeaderText="Město"
                  DataField="mesto"
            />

            <com:THyperLinkColumn
                  HeaderText="Detail"
                  Text="Detail"
                  ItemStyle.HorizontalAlign="Center"
                  DataNavigateUrlField="id">
                  <prop:DataNavigateUrlFormatString>#
                        $this->Service-
                        >constructUrl('users.DetailUser',array('id'=>{0}))
                  </prop:DataNavigateUrlFormatString>
            </com:THyperLinkColumn>

      </com:TDataGrid>

</com:TContent>
```

4.3.5. Implementace zajímavých funkcí ve frameworku Prado

Framework Prado přináší se svými komponentami a událostmi řízeným vývojem několik zajímavých funkcí, které se nejen u ostatních frameworků, ale všeobecně při vývoji webových systémů moc často nevidí. A proto si dvě zajímavé komponenty a jejich funkce podrobněji popíšeme.

Komponenta pro tvorbu průvodců nastavení (Wizard):

Komponenta *<com:TWizard>* představuje analogii pro instalační průvodce obvykle používané pro instalaci programů na systémech Windows. Tito průvodci rozdělují velké formuláře na sled menších formulářů, které po krocích vedou k dokončení instalace (nastavení). Většinou se využije standardního rozvržení oken průvodce, ale stejně tak je možné si nadefinovat vlastní uspořádání okna průvodce pomocí CSS.

Ukázka vytvoření jednoduchého průvodce nastavením:

```
<com:TWizard>
      <com:TWizardStep Title="krok 1" StepType="Start">
      Obsah kroku 1, může obsahovat řídící prvky
      </com:TWizardStep>

      <com:TWizardStep Title="krok 2" StepType="Step">
      Obsah kroku 2, může obsahovat řídící prvky
      </com:TWizardStep>

      <com:TWizardStep Title="konečný krok" StepType="Finish">
      Obsah konečného kroku, může obsahovat řídící prvky
      </com:TWizardStep>
</com:TWizard>
```

Komponenta pro tvorbu kontroly *Captcha* u formulářů:

Jako další zajímavou a především velmi důležitou komponentu nabízí framework Prado komponentu *<com:TCaptcha>*, která slouží k ověření, zda formulář vyplňuje opravdu uživatel a ne počítač. Captcha obrázky jsou dnes téměř nutností pro systémy, které jsou v prostředí internetu a umožňují ukládat například komentáře bez nutnosti přihlášení.

4.3.6. Zhodnocení

PHP framework Prado se od samého začátku jeho vývoje prezentuje jako jedinečný. Jeho jedinečné postavení na poli frameworků pro jazyk PHP si vysloužil díky přístupu, se kterým se u vývoje webových aplikací v jazyce PHP většinou nesetkáme. Jedná se o komponentový, událostmi řízený framework. Komponenta představuje malou, znovupoužitelnou softwarovou jednotku. Tyto komponenty jsou propojeny pomoci událostí, které se starají o obsloužení (implementaci) požadavku na tuto komponentu. Oproti tradičnímu použití jazyka PHP je díky komponentám vývoj zaměřen především na implementaci logiky a kód prezenční vrstvy obsahuje minimum stále se opakujících částí a stává se tak přehlednějším.

Velice dobrá je podpora systému šablon stránek. Většina stránek je rozvržena pomocí souboru kaskádových stylů a tak stačí vytvořit jeden soubor šablony s koncovkou *.tpl*, kde vložíme odkazy na

soubory se stále stejným obsahem na všech stránkách, jako hlavička, patička, nebo menu stránky. Stejně tak pomocí komponenty *<com:TContentPlaceHolder>* označíme místo pro umístění hlavního obsahu stránek. Stránka může obsahovat více obsahů, které jsou označeny příslušným identifikátorem, a v pohledu pouze určíme, ve které části se daný obsah zobrazí.

Celkové množství použitelných komponent je spousta, a proto pro efektivní použití je velmi důležitá podrobná dokumentace všech komponent, manuál a další zdroje informací pro práci s tímto frameworkem. V tomto ohledu si Prado, v porovnání s jinými PHP frameworky, vede velmi dobře a veškerý zdroj informací pro práci je k nalezení na stránkách frameworku. Je zde podrobná dokumentace všech komponent dále pak obsáhlý a aktuální manuál. Velmi užitečné jsou také videa, které usnadní a urychlí začátky s tímto frameworkem.

PHP framework Prado by mohl být dobrou volbou v případě větších projektů. Pro základní práci má minimální nároky na nastavování, ale v případě trochu náročnější aplikace, která využívá databázového systému, se bez častých zásahů do konfiguračních souborů nevyhneme. Tato skutečnost by mohla být výhodou pro větší projekty, kde je potřeba mít nad spoustou věcí kontrolu. Prado přichází se zcela novým přístupem k vývoji aplikací v jazyce PHP, který může být pro spoustu vývojářů, zvyklých na komponentový a událostmi řízený přístup, výhodou. I přes výhody, které tento framework nabízí, bych jej charakterizoval spíše jako náročnější na studium a používání. Stejně tak je jeho hardwarová náročnost v testech větší v porovnání s ostatními frameworky [22].

4.4. CodeIgniter

4.4.1. Instalace frameworku CodeIgniter

Framework CodeIgniter je distribuován jako balíček ve formátu zip. Tento soubor rozbalíme v kořenovém adresáři nainstalovaného webového serveru a přejmenujeme adresář podle jména projektu.

Struktura adresáře nově vytvořeného projektu:

```
/webroot/NazevProjektu/system    - v adresái systém jsou obsaženy všechny
                                   soubory a adresáře pro spuštění pojektu.
                                   Tedy jak knihovny a soubory frameworku,
                                   tak i námi vytvořené zdorojové soubory
                                   apilace.
                       /user_guide- zahrnuje offline verzi uživatelské
                                   příručky. Stejná uživatelská příručka je
                                   také dostupná na stránkách tohoto
                                   frameworku.
```

Požadavky na platformu:

Přestože je CodeIgniter napsaný v jazyku PHP4, tak podporuje PHP4 i PHP5. Minimální verze je 4.3.2. Vedlejší vývojová větev tohoto frameworku s názvem Kohana je programována v PHP5. CodeIgniter podporuje všechny běžně používané databázové systémy.

Již v tuto chvíli můžeme zadat do internetového prohlížeče adresu *http://localhost/NazevProjektu/* a měli bychom vidět úvodní stránku.

4.4.2. Nastavení frameworku CodeIgniter

CodeIgniter má minimální požadavky co se jeho nastavení týče. Pro základní použití stačí nastavit v adresáři */NazevProjektu/system/application/* v souboru *config.php* název URL projektu. V případě že se jedná o provoz při vývoji jenom na počítači, tak URL adresu nastavíme jako *http://localhost/NazevProjektu/*. Jinak celý název domény. Nastavení práce s databází se nachází ve stejném adresáři v souboru *database.php*. Zde se nastaví údaje přihlášení k danému databázovému serveru.

Jelikož se knihovny frameworku nacházejí ve stejném adresáři jako náš projekt, tak bych doporučil přesunout adresář *application* se soubory vytvářeného projektu do kořenového adresáře. Stejně tak se kvůli zvýšené bezpečnosti doporučuje přejmenovat název adresáře *system* jiným názvem. V případě jeho změny se musí tato skutečnost upravit v nastavení souboru *index.php*, kde musíme změnit přejmenovaný název systémové složky.

Po instalaci není nastaven ve frameworku mod_rewrite k dosažení pěkného tvaru URL adres. Proto si případně musíme vytvořit .htaccess soubor sami a umístit ho do kořenového adresáře našeho projektu. Dále pak parametr *index_page* v souboru *config.php* nastavit na prázdnou hodnotu.[15]

4.4.3. Adresářová struktura projektu

Všechny naše soubory vytvářené aplikace (kromě souborů pro vzhled stránky, obrázky apod.) jsou umístěny v adresáři *application*. Jeho vnitřní adresářová struktura je oproti jiným frameworků jednoduchá a velmi přehledná.

```
-- /config/                -obsahuje konfigurační soubory aplikace ...
   /controllers/           -obsahuje vytvořené soubory kontrolerů
   /errors/                -obsahuje vytvořené chybové stránky
   /helpers/               -rozšiřuje aplikaci o cizí helpery
   /hooks/                 -obsahuje programové kódy rozšiřující jádro
   /language/              -obsahuje soubory pro jazykovou lokalizaci
         /english/         -zahrnuje soubory překladu v konkrétním jazyce
   /libraries/             -obsahuje uživatelsky vytvořené knihovny
   /models/                -obsahuje modely vytvořené aplikace
   /views/                 -obsahuje pohledy vytvořené aplikace
```

4.4.4. Způsob práce s frameworkem a konstrukce MVC na příkladu

Implementace jednoduchého příkladu pro ukázku návrhového vzoru MVC je ve frameworku CodeIgniter jednodušší než ve všech předešlých frameworcích. CodeIgniter si klade za cíl činnosti při vývoji opravdu zjednodušit a stejné věci se pomocí něj implementují s někdy polovičním množství napsaných znaků, než kdybychom framework nepoužili a vystačili si pouze s jazykem PHP.

Návrh Modelu:

Model je v návrhovém vzoru určen pro přístup k datům uloženým v databázi. Každá databázová tabulka by měla být v aplikaci reprezentována souborem modelu. V CodeIgniter jsou součástí modelu implementované funkce pro přístup k datům, jejich mazání apod.

Pokud bychom chtěli demonstrovat jednoduchost použití tohoto frameworku, tak si vystačíme i bez implementovaného souboru modelu. Stačí pouze v kontroléru příkaz pro spojení s konkrétní databázovou tabulkou a můžeme pomocí jednoduchého dotazu data získat a v pohledu je vypsat. Tento přístup by mohl mít opodstatnění v případě aplikace, která nepřistupuje k databázi, ale i tak tento způsob narušuje principy MVC architektury.[16]

```
$data['query'] = $this->db->get('users');
$this->load->view('welcome_message', $data);
```

Měli bychom ovšem zachovávat strukturu MVC modelu a všechny jeho části implementovat. Později budeme mít strukturu modelu připravenou a můžeme přidávat do něj nové funkce.

Ukázka implementace modelu:

```
<?php
class MUzivatele extends Model {
       function MUsers () {
           parent::Model();
       }
       function Pristup($id, $data){

       }
}
?>
```

Implementace Controlleru:

Kontrolér obsluhuje požadavky a jeho funkcím jsou předávány data pomocí parametrů v příkazové řádce. Nastavení validace dat se provádí u tohoto frameworku v kontroléru. Soubory všech vytvořených kontroléru se nachází v adresáři */NazevProjektu/system/application/controllers*. Celkově by se dalo říct, že každá metoda kontroléru slouží k obsluze každého pohledu v aplikaci a každý model má přiřazený svůj kontrolér. Na celou aplikaci bychom si tedy mohli vystačit s jedním kontrolérem, který obsluhoval všechny požadavky. Pokud ovšem chceme, aby byl daný model v metodě použit, musíme jej načíst pomocí funkce *load()*.

Obsah souboru kontroléru *user.php:*

```
class User extends Controller {

      function User() {
            parent::Controller();
      }

      function index() {
        $data['titulek'] = "Databáze filmů a komentářů";
        $data['nadpis'] = "Vítejte v databázi filmů";
        $data['query'] = $this->db->get('users');
        $this->load->view('welcome_message', $data);
      }
}
```

Implementace View:

Pohledy frameworku jsou umístěny v adresáři /*NazevProjektu/system/application/views*. Mají koncovku .php a jsou to klasické HTML dokumenty do kterých je vkládán na požadovaných místech kód v jazyce PHP pro prezentaci dat. Tento způsob má nevýhodu v tom, že v případě změny vzhledu a rozvržení stránek se musí všechny pohledy v aplikaci přepsat, což může být v případě rozsáhlejší aplikace značně pracné.

Ukázka části obsahu souboru pohledu *user.php* pro vypsání tabulku uživatelů:

```
<html>
    <head>
        <title><?php echo $titulek; ?></title>
        <link href="<?=base_url()?>webroot/css/style.css" rel="stylesheet"
        type="text/css" />
    </head>

      <body>
      <div id="page">
            <h3><?php echo $nadpis; echo br(2); ?></h3>
            <?php echo $text; ?>

            <?php foreach($query->result() as $row): ?>
                  <?php echo $row->jmeno; ?>
                  <?php echo $row->prijmeni; ?>
            <?php endforeach; ?>
      </div>
      </body>
</html>
```

4.4.5. Implementace zajímavých funkcí ve frameworku CodeIgniter

Framework CodeIgniter implementuje většinu nadstandardních funkcí ve formě knihoven. Nicméně spousta těchto věcí není součástí frameworku, ale je třeba případné knihovny do frameworku přidat nakopírováním do složky s knihovnami. Tyto knihovny jsou většinou vyvíjeny komunitou okolo tohoto frameworku a dostupné na wiki stránce frameworku.

Generování koláčových grafů:

Funkce generování grafů může být velmi zajímavá pro uživatele systému díky snadné a přehledné možnosti prezentace dat uživateli.

Nastavení a možnost implementace grafů:

1) Stáhnout komprimovaný balíček s knihovnou pro implementaci grafů
2) Nakopírovat obsah balíčku do adresáře *application/libraries* našeho projektu
3) Napsání třídy kontroléru podle vzoru a předání dat příslušnému pohledu.

Implementace funkce kontroléru pro vytvoření grafu:[17]

```
$this->load->library('piechart');

// Nastavení proměnných grafu
$this->piechart->showLabel(true);
$this->piechart->showPercent(true);
$this->piechart->showParts(true);
$this->piechart->setWidth(250);
$this->piechart->setFont('c:/wamp/www/classroombookings/tahoma.ttf', 9);
$this->piechart->setLegend('round');
$this->piechart->setData( array(14,5,11,10,20) );
$this->piechart->setLabels(array('Jablka','Hrušky','Třešně','Jahody',
'Maliny') );

// Vytvoříme jedinečný název souboru
$hash = md5("report-pie-".$this->school_id);

// Vygenerování a uložení grafu
$this->piechart->Generate("webroot/images/reports/$hash.png");

$layout['title'] = 'Výsledky';
$layout['showtitle'] = $layout['title'];
$layout['body'] = $this->load->view('reports/reports_index', NULL, True);
$layout['body'] .= '<img src="webroot/images/reports/'.$hash.'.png">';

$this->load->view('layout', $layout);
```

4.4.6. Zhodnocení

CodeIgniter je jednoduchý a efektivní framework a jeho nastudování je rychlejší. To znamená, že můžete být produktivní v tomto frameworku v relativně kratší dobu oproti ostatním. Stejně tak jeho adresářová struktura je přehledná a působí „čistějším“ dojmem, kde zde není spousta prázdných a nevyužívaných adresářů. Nevyžaduje omezující pravidla a konvence pro psaní programu, ale i přitom je jeho kód přehledný a jasný. Osahuje velké množství tříd pro práci, které se dají velmi jednoduchým způsobem integrovat a používat v aplikaci.

Jako další přednost bych viděl to, že za tímto frameworkem stojí firma a ne komunita. Pokud se rozhodneme použít daný framework pro komerční účely a bude v produkčním prostředí delší dobu, tak je velmi důležitá další efektivní podpora, rozvoj a oprava chyb. V případě komunity jsou tyto skutečnosti nejisté. Přesto, že za CodeIgniter stojí firma, tak je postaven na open source filozofii a je k dispozici zcela zdarma.

Největším nedostatkem frameworku CodeIgniter je absence struktury pro vzhled a šablony webové stránky. Předem vytvořené soubory pro kaskádové styly CSS, obrázky anebo soubory JavaScriptu zde nejsou a ani se na ně nedá jednoduchým příkazem odkazovat. Proto si musíme takovouto strukturu sami vytvořit a pouze předpokládat, že její umístění ve struktuře frameworku bude fungovat. PHP framework, jakožto struktura pro tvorbu webových systému by takovouto vlastnost jednoznačně měla mít v sobě standardně zahrnutou.

5. Závěr

Tato bakalářská práce měla za úkol zhodnotit situaci frameworků pro programovací jazyk PHP. Webové aplikace stejně jako skriptovací jazyk PHP jsou čím dál více ve větší oblibě, a proto by se při vzniku nového systému mělo dbát na zvážení všech případných požadavků, které od vybraného frameworku očekáváme. Výběr toho správného frameworku není jednoduchý a určitě je zapotřebí zvážit všechny jeho možnosti pro daný projekt a vývojáře, protože čas vynaložený na studium není určitě zanedbatelný. Dalo by se říci, že má smysl použít framework pro každý nový projekt pokud se nejedná o webové stránky se statickým obsahem případně s páru standardními funkcemi. K výběru je skutečně veliké množství frameworků, ale právě tato situace může být spíše nevýhodou, nebo většinou jedná pouze o malé projekty, které nepokrývají problematiku vývoje webových komplexně. Další slabinu vidím ve značné roztříštěnosti řešení, s jakým každý framework přichází. Každý má zcela individuální přístup pro řešení daného problému. Tato skutečnost může být způsobena absencí velké organizace, která by za jazykem PHP stála a určovala jednotný základ, který můžeme vidět u frameworku .NET nebo okolo jazyka Java.

Seznam použité literatury

[1] *Framework* [online]. 2008 [cit. 2008-11-12]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Framework>.

[2] *Web application framework* [online]. 2008 [cit. 2008-11-12]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Web_application_framework>.

[3] *PHP Frameworks* [online]. c2007 [cit. 2008-11-23]. Dostupný z WWW: <http://www.phpframeworks.com>.

[4] AZAD, Kalid. *Intermediate Rails: Understanding Models, Views and Controllers* [online]. 2007 [cit. 2008-11-12]. Dostupný z WWW: <http://betterexplained.com/articles/intermediate-rails-understanding-models-views-and-controllers/>.

[5] AHSANUL, Bari, ANUPOM, Syam. *CakePHP Application Development*. [s.l.] : [s.n.], 2008. 311 s. ISBN 978-1-847193-89-6.

[6] GRUDL, David. *Texy!* [online]. c2008 [cit. 2008-11-23]. Dostupný z WWW: <http://texy.info/cs/>.

[7] BAKER, Marcus. *Simple Test for PHP* [online]. c2003 [cit. 2008-11-25]. Dostupný z WWW: <http://www.lastcraft.com/simple_test.php>.

[8] *CakePHP: the rapid development php framework* [online]. [2008] [cit. 2008-12-29]. Dostupný z WWW: <http://cakephp.org>.

[9] *Symfony | Web PHP Framework* [online]. [2008] [cit. 2008-12-29]. Dostupný z WWW: <http://www.symfony-project.org>.

[10] *PRADO PHP Framework* [online]. c2004-2008 [cit. 2008-12-29]. Dostupný z WWW: <http://www.pradosoft.com>.

[11] *CAPTCHA* [online]. 2009 [cit. 2009-01-30]. Dostupný z WWW: <http://en.wikipedia.org/wiki/CAPTCHA>.

[12] *CodeIgniter - Open source PHP web application framework* [online]. c2001-2008 [cit. 2008-12-29]. Dostupný z WWW: <http://codeigniter.com>.

[13] CHAN, Kai, OMOKORE, John, K. MILLER, Richard. *Practical CakePHP Projects*. [s.l.] : [s.n.], 2009. 389 s. ISBN 978-1-4302-1579-0.

[14] *ISO 639-2 Language Code List - Codes for the representation of names of languages (Library of Congress)* [online]. 2008 [cit. 2009-03-03]. Dostupný z WWW: <http://www.loc.gov/standards/iso639-2/php/code_list.php>.

[15] MYER, Thomas. *Professional CodeIgniter*. [s.l.] : [s.n.], 2008. 314 s. ISBN 978-0-470-28245.

[16] UPTON, David. *CodeIgniter for Rapid PHP Application Development*. [s.l.] : [s.n.], 2007. 244 s. ISBN 978-1-847191-74-8.

[17] *3D Pie Chart Library* [online]. [2008] [cit. 2009-03-10]. Dostupný z WWW: <http://codeigniter.com/wiki/3d-pie-chart/>.

[18] XUE, Qiang, ZHUO, Wei. *PRADO v3.1.4 Quickstart Tutorial*. [s.l.] : [s.n.], 2009. 272 s. Dostupný z WWW: <http://www.pradosoft.com/docs/quickstart.pdf>.

[19] *Active record pattern* [online]. 2009 [cit. 2009-03-26]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Active_record_pattern>.

[20] ZANINOTTO, François, POTENCIER, Fabien. *The Definitive Guide to symfony*. [s.l.] : [s.n.], 2007. 486 s. ISBN 978-1-59059-786-6.

[21] *Symfony | Web PHP Framework* [online]. [2009] [cit. 2009-04-08]. Dostupný z WWW: <http://www.symfony-project.org/>.
[22] *Velký test PHP frameworků - Root.cz* [online]. 2008 [cit. 2009-04-07]. Dostupný z WWW: <http://www.root.cz/clanky/velky-test-php-frameworku-2008/>.

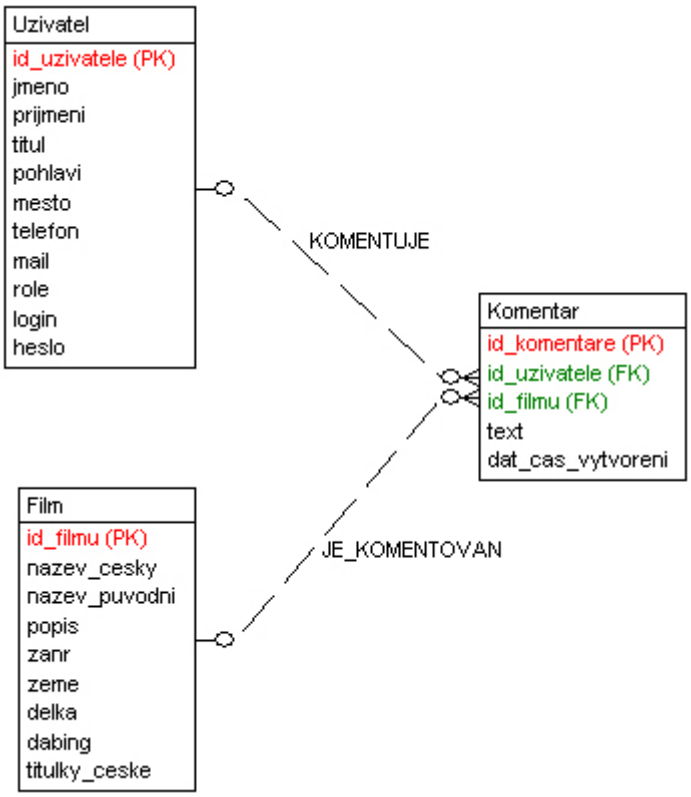
Přílohy

I. Ukázkový systém v jednotlivých PHP frameworcích

Aby byl popis a zhodnocení vybraných frameworků objektivní, tak bude v každém z těch frameworků implementován jednoduchý systém pro vytváření komentářů k filmům. Tento systém by měl obsahovat základní funkce, které se v internetových aplikacích vyskytují. Především obsluhu databázových tabulek a všechny základní funkce s ní spojené jako vytváření, editace a mazání záznamů. Stejně tak nastavení PHP frameworku, pokud je součástí zdrojových kódů celé aplikace.

Datová Analýza

ER diagram



Lineární zápis typů entit

Uzivatel(id_uzivatele, jmeno, prijmeni, titul, pohlavi, město, telefon, mail, role, login, heslo)

Film(id_filmu, nazev_cesky, nazev_puvodni, popis, zanr, zeme, delka, dabing, titulky_ceske)

Komentar(id_komentare, id_uzivatele, id_filmu, text, dat_cas_vytvoreni)

Lineární zápis typů vazeb

KOMENTUJE(Uzivatel, Komentar)

MA_KOMENTAR(Komentar, Uzivatel)

Výše uvedená datová analýza je pouze ukázkou jak by mohla vypadat. Skutečná implementace datového modelu v databázi se může lišit a to především v názvech typů entit a jejich atributů. Tato skutečnost je způsobena především tím, že některé PHP frameworky a jejich obecné zásady pro vývoj v nich si vyžadují konvence pro pojmenování typů entit a názvů atributů.

II. Obsah přiloženého CD

Adresář	**Obsah**
`/text/`	Elektronická verze této práce
`/frameworky/`	Instalační soubory použitých frameworků
`/ukazky/`	Ukázky systému
`/ukazky/cakephp/`	Implementace systému v PHP frameworku CakePHP
`/ukazky/symfony/`	Implementace systému v PHP frameworku Symfony
`/ukazky/prado/`	Implementace systému v PHP frameworku Prado
`/ukazky/codeigniter/`	Implementace systému v PHP frameworku CodeIgniter
`/sablona/`	HTML, CSS šablona systému a obrázky uživatelského rozhraní